Iterative Decoding of Serially Concatenated Codes with Interleaves and Comparison with Turbo Codes

S. Benedetto, G. Montorsi
Dipartimento di Elettronica, Politecnico di Torino
C.so Duca degli Abruzzi n.24
10129 Torino, Italy.

D. Divsalar, F. Pollara
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109- USA.

Abstract - *A serially concatenated code with interleaver consists of the* cascade of *an* outer *encoder, an interleaver permuting the outer codewords bits, and an inner encoder whose input words are the permuted outer codewords. We propose a new, low-complexity it erat ive* decoding *algorithm for* serially *concatenated codes and apply it to perform simulation comparisons with parallel* concatenated con*volutional codes known as " turbo codes".*

I. Introduction

As an alternative to the "turbo" codes [1], which are formed by two parallel concatenated convolutional encoders, such that the input bits to the second encoder are the scrambled version (through an interleave) of those entering the first encoder, we have proposed in [2;3] the serial concatenation of interleaved codes or *serially concatenated codes* (SCCs). It consists (see the upper part of Figure 1) of the cascade of an *outer* encoder and an *inner* encoder joined by an inter-leaver, denoted as π.

Analytical upper bounds to the performance of a maximum-likelihood (ML) decoder for SCC have been presented in [2], whereas design guidelines leading to the optimal choice of the CCs that maximize the *interleaver gain* and the asymptotic code performance have been included in [3].

The conclusion of the previously mentioned analysis and design was very promising, in the sense that the interleaver gain of serially concatenated convolutional codes (SCCCs), defined as the inverse of the factor (function of the interleave length N) by which the bit error probability decreases, can be significantly larger than for turbo codes. This result, however, had been derived for concatenated codes employing the *uniform* interleaver defined in [4] and decoded using an ML algorithm, which is known to be exceedingly complex for medium-large interleaves.

In this paper, we extend the previous results to the practical case of low complexity decoding algorithms. First, we present a new iterative decoding algorithm yielding results close to capacity limits. Then, we apply the decoding algorithm to simulate the behavior of several SCCCs, and, finally, we perform comparisons with turbo codes of the same complexity and decoding delay, With this embodiment of results, we believe that SCCC can be considered as a valid, in some cases superior, alternative to turbo codes.

The research in this paper was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration (NASA), and at the Politecnico di Torino. The research was also partially supported by NATO under Research Grant CRG 951208 and Agenzia Spaziale Italiana (ASI).

II. Iterative decoding of serially concatenated codes

In this section, we present a new iterative algorithm for decoding serially concatenated codes, with complexity not significantly higher than that needed to separately decode the two CCs. Because of the importance in applications, all examples will refer to SCCCs, although the decoding algorithm can be applied to serially concatenated block codes as well.

The core of the new decoding procedure consists of a block called SISO (Soft-input Soft-Output). It is a four-port device, which accepts as inputs the probability distributions (or the corresponding likelihood ratios) of the information and code symbols labeling the edges of the code trellis, and forms as outputs an update of these probability distributions based upon the code constraints. The block SISO is used within the iterative decoding algorithm as shown in Figure 1, where we also show the block diagram of the encoder to clarify the notations.

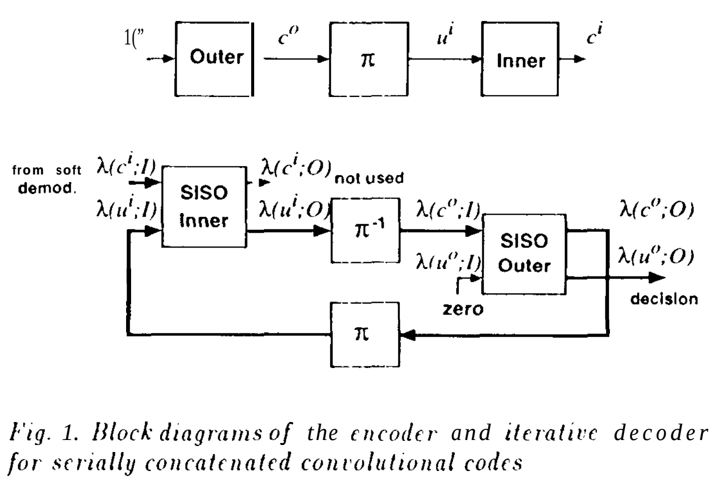

Fig. 1. Block diagrams of the encoder and iterative decoder for serially concatenated convolutional codes

We will first explain in words how the algorithm works, according to the blocks of Figure 1. Successively, we will give the input-output relationships of the block SISO.

The symbols $\lambda(\cdot;I)$ and $\lambda(\cdot;O)$ at the input and output ports of SISO refer to the logarithmic likelihood ratios

(LLRs)', unconstrained when the second argument is 1, and modified according to the code constraints when it is O. The **first** argument u refers to the information symbols of the encoder, whereas c refers to code symbols. Finally, the superscript o refers to the outer encoder, arid i to the inner encoder. The LLRs are defined as

$$\lambda(x;\ \cdot) \triangleq \log\left[\frac{P(x;\cdot)}{P(x_{\mathrm{ref}};\cdot)}\right] \tag{1}$$

When x is a binary symbol, 'LO" or "1", x_{ref} is generally assumed to be the "1". When x belongs to an L-ary alphabet, we can choose as x_{ref} each one of the L symbols; a common choice for hardware implementation is the symbol with the highest probability, so that one LLR will be equal to zero and all others negative numbers.

Differently from the iterative decoding algorithm employed for turbo decoding, in which only the LLRs of information symbols are updated, we must update here the LLRs of both information and code symbols based on the code constraints.

During the first iteration of the SCCC algorithm, the block "SISO Inner" is feel with the demodulator soft outputs, consisting of the LLRs of symbols received from the channels, i.e. of the code symbols of the inner encoder. The second input $\lambda(u^i;I)$ of the S1S0 Inner is set to zero during the first iteration, since no a-priori information is available on the input symbols u^i of the inner encoder.

The LLRs $\lambda(c^i;I)$ are processed by the S1S0 algorithm, which computes the *extrinsic* LLRs of the information symbols of the inner encoder $\lambda(u^i;O)$ conditioned on the inner code constraints. The extrinsic I, I, Its are passed through the inverse interleaver (block labeled "π^{-1}"), whose outputs correspond to the I, LRs of the code symbols of the outer code, i.e.

$$\pi^{-1}[\lambda(u^i;O)] = \lambda(c^o;I)$$

These LLRs are then sent to the block "SISO Outer" in its upper entry, which corresponds to code symbols. The S1S0 Outer, in turn, processes the LLRs $\lambda(c^o;I)$ of its unconstrained code symbols, and computes the LLRs of both code and information symbols based on the code constraints. The input $\lambda(u^o;I)$ of the S1S0 Outer is always set to zero, which implies assuming equally likely transmitted source information symbols. The output LLRs of information symbols (which yield the a-posteriori LLRs of the SCCC information symbols) will he used in the final iteration to recover the information hits. On the other hand, the LLRs of outer code symbols, after interleaving are fed hack to the lower entry (corresponding to information symbols of the inner code) of the block S1S0 inner to start. the second iteration. In fact we have

$$\pi[\lambda(c^o;O) = \lambda(u^i;I)$$

A. *The input-output relationships for the block SISO*

The block S[S0 has been described in [5]. It represents a slight generalization of the BCJR algorithm (see [6;7]. here, we will only recall for completeness its input-output relationships. They will refer, for notations, to the trellis section of the trellis encoder shown in Figure 2, where the symbol e denotes the trellis edges, and where we have identified the information and code symbols associated to the edge e as $u(e), c(e)$, and the starting and ending states of the edge e as $s^S(e), s^E(e)$, respectively.

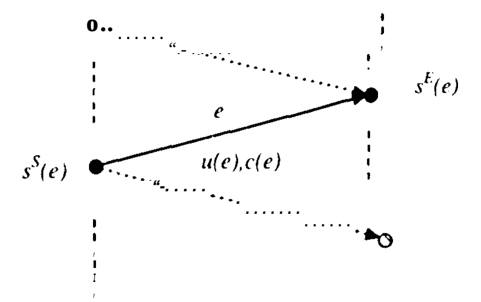

Fig. 2. Trellis section defining the notations used for the description of the SISO algorithm

The block S1S0 works at *symbol* level, i.e., for an (n, p) convolutional code, it operates on information symbols u belonging to an alphabet with size 2^p and on code symbols belonging to an alphabet wit b size 2^n. We will give the general input-output relationships, valid for both outer and inner S1S0s, assuming that the in formation and code symbols are defined over a finite time index set [1,...,1{].

At time k, $k = 1,\ldots,1<$, the output extrinsic LLRs are computed as

$$\begin{aligned}\lambda_k(c;\ O) &= \max_{e:c(e)=c}{}^*\{\alpha_{k-1}[s^S(e)] + \lambda_k[u(e);\ I] \\ &\quad + \beta_k[s^E(e)]\} + h_c\end{aligned} \tag{2}$$

$$\begin{aligned}\lambda_k(u;\ O) &= \max_{e\ u(e)=u}{}^*\{\alpha_{k-1}[s^S(e)] + \lambda_k[c(e);I] \\ &\quad + \beta_k[s^E(e)]\} + h_u\end{aligned} \tag{3}$$

The name *extrinsic* given to the LLRs computed according to (2) and (3) derives from tile fact that the evaluation of $\lambda_k(c;O)$ (and of $\lambda_k(u;O)$) *does* not depend on the corresponding simultaneous input $\lambda_k(c;I)$ (arid $\lambda_k(u;I)$), so that it can he considered as an update of the input LLR based 011 informations coming from all homologous symbols in the *sequence*, except the one corresponding to the same symbol interval.

The quantities $\alpha_k(\cdot)$ and $\beta_k(\cdot)$ in (2) and (3) arc obtained through the *forward* and *backward* recursions, respectively, as

$$\begin{aligned}\alpha_k(s) &= \max_{e:s^E(e)=s}{}^*\{\alpha_{k-1}[s^S(e)] + \lambda_k[u(e);I] + \\ &\quad + \lambda_k[c(e);I]\},\ k = 1,\ldots,K-1 \\ \beta_k(s) &= \max_{e:s^S(e)=s}{}^*\{\beta_{k+1}[s^E(e)] + \lambda_{k+1}[u(e);I] +\end{aligned} \tag{4}$$

[1] When the symbols are binary, only one LLR is needed, when the symbols belong to an L-ary alphabet, $L-1$ LLRs are required

$$+\lambda_{k+1}[c(\cdot);I]\},\ k=K-1,\ldots,1, \quad (5)$$

with suitable initial values. The quantities h_c, h_u are normalization constants.

The operator max* performs the following operation

$$\max_j{}^*(a_j) \triangleq \log\left[\sum_{j=1}^{J} e^{a_j}\right] \quad (6)$$

which, in practice, can be performed as

$$\max_j{}^*(a_j) = \max_j(a_j) + \delta(a_1, a_2, \ldots, a_J) \quad (7)$$

where $\delta(a_1, a_2, \ldots a_J)$ is a correction term that can be computed recursively using a single-entry look-up table [8;9].

The previous description of the iterative decoder assumed that all operations were performed at *symbol* level. Quite often, however, the interleave operates at *bit* level to be more effective. Thus, to perform bit interleaving, we need to transform the symbol extrinsic LLRs obtained at the output of the first S1S0 into extrinsic bit LLRs, before they enter the deinterleaver. After deinterleaving, the bit LLRs need to be compacted into symbol LLRs before entering the second S1S0 block, and so on.

These operations are performed under the assumption that the bits forming a symbol are independent.

Assuming an (n, p) code, and denoting with $\mathbf{u} = [u_1, \ldots, u_p]$ the information symbol formed by p information bits, the the extrinsic LLR λ_i of the i-th bit u_i within the symbol $\mathbf{u}$ is obtained as

$$\begin{aligned}\lambda_i(u;O) &= \max_{\mathbf{u}:u_i=u}{}^*[\lambda_k(\mathbf{u};O)+\lambda(\mathbf{u};I)]+\\ &- \max_{\mathbf{u}:u_i=1}{}^*[\lambda(\mathbf{u};O)+\lambda_k(\mathbf{u};I)]-\lambda_i(u;I)\end{aligned}$$

Conversely, the extrinsic LLR of the symbol $\mathbf{u}$ is obtained from the extrinsic LLRs of its component bits u_i as

$$A(n) = \sum_{i=1}^{F} \lambda_i(u) \quad (8)$$

As previous description should have made clear, the S1S0 algorithm requires that the whole sequence had been received before starting. The reason is due to the backward recursion that starts from the (supposed known) final trellis state. A more flexible decoding strategy is offered by modifying the algorithm in such a way that the S1S0 module operates on a fixed memory span, and outputs the smoothed probability distributions after a given delay D. This algorithm, which we have called the *sliding window soft-input soft-output (SW-S1S0)* algorithm, is fully described in [9]. To obtain the following simulation results, the SW-SISO algorithm has been applied.

III. APPLICATIONS OF THE DECODING ALGORITHM

We will now use the decoding algorithm to confirm the design rules presented in [3], and to show the behavior of SCCC

Code description	$G(D)$
Rate 1/2 R	$\left[1, \frac{1+D^2}{1+D+D^2}\right]$
Rate 1/2 NR	$\left[1+D+D^2, \ 1+D^2\right]$
Rate 2/3 R	$\begin{bmatrix}1, & 0, & \frac{1+D^2}{1+D+D^2}\\ 0, & 1, & \frac{1+D}{1+D+D^2}\end{bmatrix}$
Rate 2/3 NR	$\begin{bmatrix}1+D, & D, & 1\\ 1+D, & 1, & 1+D\end{bmatrix}$

Table 1. Generating matrices for the constituent convolutional codes

Code	Outer code			Inner code			
	Code	w_m^o	d_f^o	Code	w_m^i	d_f^i	$d_{f,eff}^i$
SCCC1	1/2 R	2	5	2/3 NR	1	3	4
SCCC2	1/2 NR	1	5	2/3 R	2	3	4

Code	SCCC			
	h_m	$\alpha(h_m)$	$h(\alpha_M)$	α_M
SCCC1	5	-4		
SCCC2	5	-4	7	-3

Table 2 Design parameters of CCs and SCCCs for two SCCCs

in the region of low signal-to-noise ratios (below cutoff rate), where analytical bounds fail to give significant results. In all simulations performed to obtain the results that we will present in the following, we have used randomly chosen convolutional interleaves and continuous decoding. The three SCCCs employed in the simulations are described, with their main parameters (see [3] for their meaning), in Table 2. They use combinations, as outer and inner codes, of four different CCs whose generating matrices are reported in Table 1.

A. The effect of a non recursive inner encoder

The analysis in [3] came to the conclusion that a non recursive inner encoder should yield little interleave gains. To confirm this theoretical prediction by simulation results, we plot in Fig. 3 the bit error probability versus the input decoding delay obtained by simulating the concatenated code SCCC1 of Table 2. This code uses as inner encoder a 4-state non recursive encoder. The curves refer to a signal-to-noise ratio $E_b/N_0 = 1.5$ dB, and to a number of iterations N_I ranging from 1 to 10. It is evident that the bit error probability reaches the floor of 10^{-5} for a decoding delay greater than or equal to 1024, so that no interleaver gain takes place beyond this point. For comparison, we report in Fig. 4 the results obtained for the code SCCC2 of Table 3. The curves refer to a signal-to-noise ratio of 0.75 dB, and show the interleaver gain predicted by the analysis.

B. Approaching the theoretical Shannon limity

We discuss here the capabilities of SCCCs of yielding results close to the Shannon capacity limit. To this purpose, we have chosen a rate 1/4 concatenated scheme with very long interleaver, corresponding to an input decoding delay of 16,384. The constituent codes are 8-state codes: the outer encoder is non recursive, and the inner encoder is a recursive encoder.

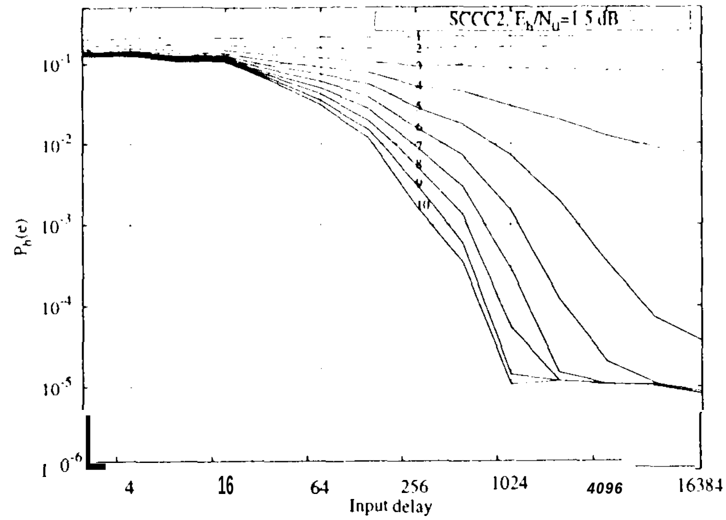

Fig. **3.** *Simulated performance of concatenated code SCCC1 of Table 2. The bit error probability is plotted versus input decoding delay for different number of iterations. The signal-to-noise ratio is 1.5 dB.*

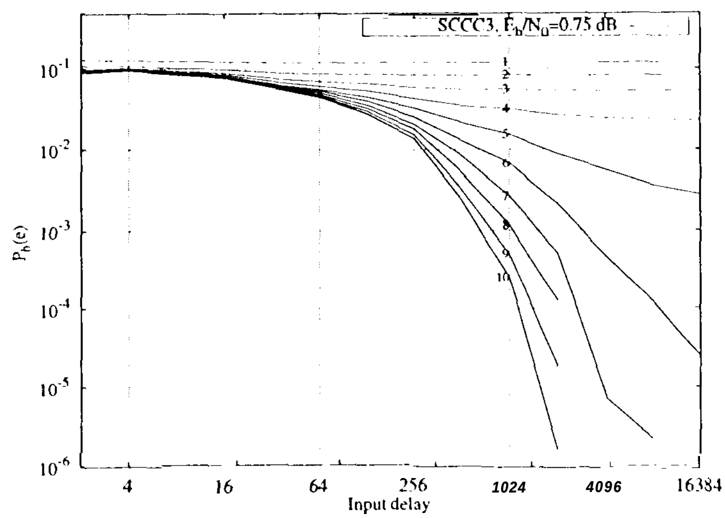

Fig. 4. Simulated performance of concatenated code SCCC2 of Table 2. The bit error probability is plotted versus input decoding delay for different number of iterations. The signal-to-noise ratio is 0.75 dB.

Their generating matrices are

$$G_o(D) = [1 + D,\ 1 + D + D^3]$$

$$G_i(D) = [1, \frac{1 + D + D^3}{1 + D}] ,$$

respectively. Note the feedback polynomial $(1 + D)$ of the inner encoder, which eliminates error events with odd input weights. The results in terms of bit error probability versus signal-to-noise ratio for different number of iterations are presented in Fig. 5. They show that the decoding algorithm works at $E_b/N_0 = -0.05$ dB, at 0.75 dB from the Shannon capacity limit, with very limited complexity (remember that we are using two rate 1/2 codes with 8 states).

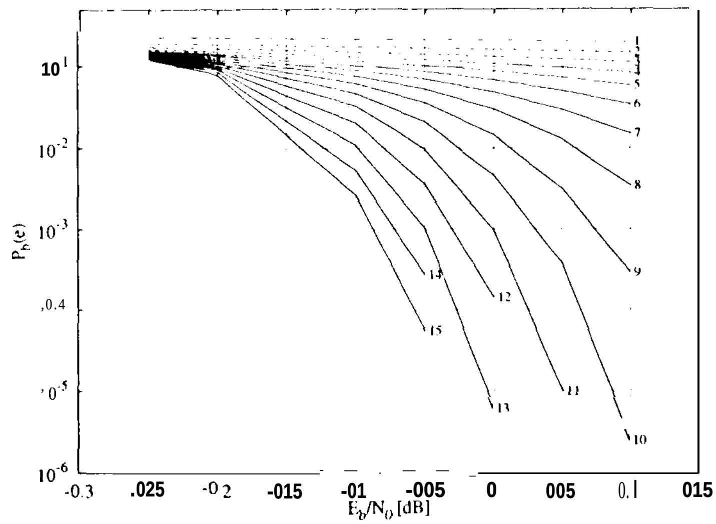

Fig. 5. Simulated performance of a rate 1/4 serially concatenated code obtained with two eight-state CCs and an interleaver yielding an input decoding delay equal to 16384.

C. Comparison between serially and parallel concatenated codes

To check if the advantages of SCCC over turbo codes predicted by the analysis are retained when the codes are iteratively decoded at very low signal-to-noise ratios, we have simulated the behavior of SCCCs and PCCCs in equal system conditions: the concatenated code rate is 1/3, the CCs are 4-state recursive encoders (rates 1/2 + 1/2 for PCCCs, and rates 1/2 + 2/3 for the SCCCs), and the decoding delays in terms of input bits are 1024 and 16,384.

In Fig. 6 we report the results, in terms of bit error probability versus signal-to-noise ratio, for the case of a decoding delay equal to 1024, after three and seven decoding iterations. As it can be seen from the curves, the PCCC outperform the SCCC for high values of the bit error probabilities. For bit error probabilities lower than 10^{-2}, the SCCC outperforms the PCCC. In particular, we notice the absence of error floor[2] in the SCCC performance. At 10^{-4}, SCCC has an advantage of 0.7 dB with seven iterations. Finally, in Fig. 7, we report the results for an input decoding delay of 16,384 and six and nine decoding iterations. In this case, the crossover between PCCC and SCCC happens around 10^{-5}. The advantage of SCCC at 10^{-6} is 0.5 dB with nine iterations.

As a conclusion, we can say that the advantages obtained for signal-to-noise ratios above the cutoff rate, where the

[2] It is customary to call "error floor" what is actually a sensible change of slope of the performance curve

union bounds can be safely applied [3], are retained also in the region between channel capacity and cutoff rate. Only when the system interest focuses on high values of bit error probability (the threshold depending on the interleaver size) the PCCC are to be preferred. PCCCs, however, present a floor to the bit error probability, which, in the most favourable case seen above, lies around 10^{-6}. This floor is absent, or, at least, much lower, in the case of SCCC.

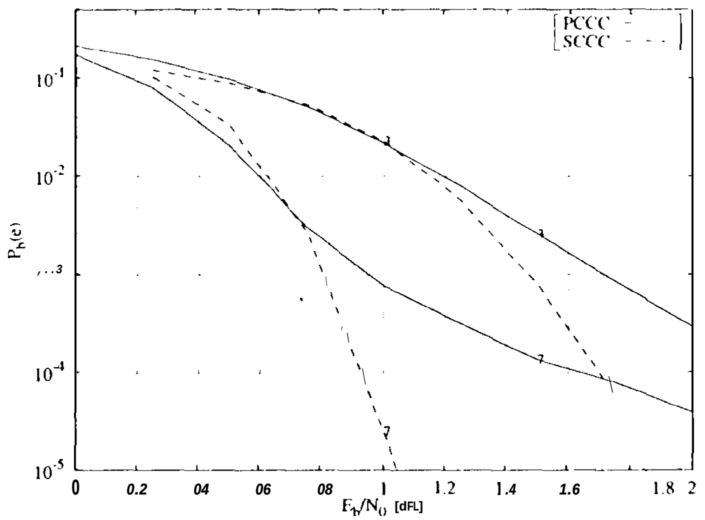

Fig. 6. Comparison of two rate 1/3 PCCC and SCCC. The PCCC is obtained concatenating two equal rate 1/2 4 states codes (first code in Table 1); the SCCC is the code SCCC1 of Table 2. The curves refer to three and seven iterations of the decoding algorithm and to an equal input decoding delay of 1024.

IV. CONCLUSIONS

An iterative decoding algorithm for serially concatenated codes with interleaver has been proposed and applied to various code configurations. Extensive simulation results have been presented, and comparisons with parallel concatenated convolutional codes have been performed, showing that the new schemes can often yield superior performance.

REFERENCES

[1] Claude Berrou, Alain Glavieux, and Punya Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", in *Proceedings of ICC '93*, Geneve, Switzerland, May 1993, pp. 1064-1070.

[2] Sergio Benedetto and Guido Montorsi, "Serial concatenation of interleaved codes: analytical performance bounds", in *Proceedings of GLOBECOM '96*, London, UK, Nov. 1996.

[3] Sergio Benedetto, Dariush Divsalar, Guido Montorsi, and Fabrizio Pollara, "Design of serially concatenated interleaved codes", in *Proceedings of ICC'97*, Montreal, Canada, June 1997.

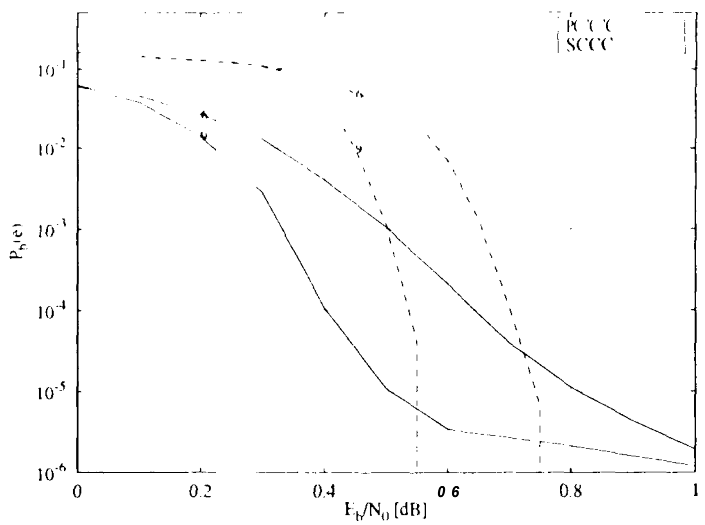

Fig. 7. Comparison of two rate 1/3 PCCC and SCCC. The PCCC is obtained concatenating two equal rate 1/2 4 states codes (first code in Table 1); the SCCC is the code SCCC2 of Table 2. The curves refer to six and nine iterations of the decoding algorithm and to an equal input decoding delay of 16384.

[4] Sergio Benedetto and Guido Montorsi, "Unveiling turbo-codes: some results on parallel concatenated coding schemes", *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 409429, Mar. 1996.

[5] Sergio Benedetto, Dariush Divsalar, and Fabrizio Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes", *IEEE Communications Letters*, vol. 1, no. 1, pp. 22-24, Jan. 1997.

[6] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transactions on Information Theory*, pp. 284-287, Mar. 1974.

[7] R.J. McEliece, "On the BCJR trellis for linear block codes", *IEEE Transactions on Information Theory*, vol. IT-42, pp. 1072-1091, July 1996.

[8] Patrick Robertson, Emmanuelle Villebrun, and Peter Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain", in *Proceedings of ICC'95*, Seattle, Washington, June 1995, pp. 1009-1013.

[9] Sergio Benedetto, Dariush Divsalar, Guido Montorsi, and Fabrizio Pollara, "Soft-input soft-output building blocks for the construction and distributed iterative decoding of code networks", *European Transactions on Telecommunications*, invited paper, to be published, May 1997.